MYSQL规约

# 建表规约

1. 表达是与否概念的字段，必须使用 is\_xxx 的方式命名，数据类型是 unsigned tinyint

（ 1 表示是，0 表示否），此规则同样适用于 odps 建表。

说明：任何字段如果为非负数，必须是 unsigned。

1. 表名、字段名必须使用小写字母或数字；禁止出现数字开头，禁止两个下划线中间只出现数字。数据库字段名的修改代价很大，因为无法进行预发布，所以字段名称需要慎重考虑。

正例：getter\_admin，task\_config，level3\_name

反例：GetterAdmin，taskConfig，level\_3\_name

1. 表名不使用复数名词。

说明：表名应该仅仅表示表里面的实体内容，不应该表示实体数量，对应于 DO 类名也是单数形式，符合表达习惯。

1. 禁用保留字，如 desc、range、match、delayed 等，参考官方保留字。
2. 唯一索引名为 uk\_字段名；普通索引名则为 idx\_字段名。

说明：uk\_ 即 unique key；idx\_ 即 index 的简称。

1. 小数类型为 decimal，禁止使用 float 和 double。

说明：float 和 double 在存储的时候，存在精度损失的问题，很可能在值的比较时，得到不正确的结果。如果存储的数据范围超过 decimal 的范围，建议将数据拆成整数和小数分开存储。

1. 如果存储的字符串长度几乎相等，使用 CHAR 定长字符串类型。
2. varchar 是可变长字符串，不预先分配存储空间，长度不要超过 5000，如果存储长度大于此值，定义字段类型为 TEXT，独立出来一张表，用主键来对应，避免影响其它字段索引效率。
3. 表必备三字段：id, gmt\_create, gmt\_modified。

说明：其中 id 必为主键，类型为 unsigned bigint、单表时自增、步长为 1；分表时改为从TDDL Sequence 取值，确保分表之间的全局唯一。gmt\_create, gmt\_modified 的类型均为date\_time 类型。

1. 表的命名最好是加上“业务名称\_表的作用”，避免上云梯后，再与其它业务表关联时有混淆。

正例：tiger\_task / tiger\_reader / mpp\_config

1. 库名与应用名称尽量一致。
2. 如果修改字段含义或对字段表示的状态追加时，需要及时更新字段注释。
3. 字段允许适当冗余，以提高性能，但是必须考虑数据同步的情况。冗余字段应遵循：

1）不是频繁修改的字段。

2）不是 varchar 超长字段，更不能是 text 字段。

正例：各业务线经常冗余存储商品名称，避免查询时需要调用 IC 服务获取。

1. 单表行数超过 500 万行或者单表容量超过 2GB，才推荐进行分库分表。

说明：如果预计三年后的数据量根本达不到这个级别，请不要在创建表时就分库分表。

反例：某业务三年总数据量才 2 万行，却分成 1024 张表，问：你为什么这么设计？答： 1024分张表，不是标配吗？

1. **合适的字符存储长度，不但节约数据库表空间、节约索引存储，更重要的是提升检索速度。**

正例：人的年龄用 unsigned tinyint（表示范围 0-255，人的寿命不会超过 255 岁）；海龟就必须是 smallint，但如果是太阳的年龄，就必须是 int；如果是所有恒星的年龄都加起来，那么就必须使用 bigint。

# 索引规约

1. 业务上具有唯一特性的字段，即使是组合字段，也必须建成唯一索引。

说明：不要以为唯一索引影响了 insert 速度，这个速度损耗可以忽略，但提高查找速度是明显的；另外，即使在应用层做了非常完善的校验和控制，只要没有唯一索引，根据墨菲定律，必然有脏数据产生。

1. 超过三个表禁止 join。需要 join 的字段，数据类型保持绝对一致；多表关联查询时，保证被关联的字段需要有索引。

说明：即使双表 join 也要注意表索引、SQL 性能。

1. 在 varchar 字段上建立索引时，必须指定索引长度，没必要对全字段建立索引，根据实际文本区分度决定索引长度。

说明：索引的长度与区分度是一对矛盾体，一般对字符串类型数据，长度为 20 的索引，区分度会高达 90%以上，可以使用 count(distinct left(列名, 索引长度))/count(\*)的区分度来确定。

1. **页面搜索严禁左模糊或者全模糊，如果需要请走搜索引擎来解决。**

说明：索引文件具有 B-Tree 的最左前缀匹配特性，如果左边的值未确定，那么无法使用此索引。

1. 如果有 order by 的场景，请注意利用索引的有序性。order by 最后的字段是组合索引的一部分，并且放在索引组合顺序的最后，避免出现 file\_sort 的情况，影响查询性能。

正例：where a=? and b=? order by c; 索引：a\_b\_c

反例：索引中有范围查找，那么索引有序性无法利用，如：WHERE a>10 ORDER BY b; 索引a\_b无法排序。

1. **利用覆盖索引来进行查询操作，来避免回表操作。**

说明：如果一本书需要知道第 11 章是什么标题，会翻开第 11 章对应的那一页吗？目录浏览一下就好，这个目录就是起到覆盖索引的作用。

正例：IDB 能够建立索引的种类：主键索引、唯一索引、普通索引，而覆盖索引是一种查询的一种效果，用 explain 的结果，extra 列会出现：using index.

1. **建组合索引的时候，区分度最高的在最左边。**

正例：如果 where a=? and b=? ，a 列的几乎接近于唯一值，那么只需要单建 idx\_a 索引即可。

说明：存在非等号和等号混合判断条件时，在建索引时，请把等号条件的列前置。 where a>?。如：and b=? 那么即使 a 的区分度更高，也必须把 b 放在索引的最前列。

1. 创建索引时避免有如下极端误解：

1）误认为一个查询就需要建一个索引。

2）误认为索引会消耗空间、严重拖慢更新和新增速度。

3）误认为唯一索引一律需要在应用层通过“先查后插”方式解决。

# SQL 规约

1. 不要使用 count(列名)或 count(常量)来替代 count(\*)，count(\*)就是 SQL92 定义的标准统计行数的语法，跟数据库无关，跟 NULL 和非 NULL 无关。

说明：count(\*)会统计值为 NULL 的行，而 count(列名)不会统计此列为 NULL 值的行。

1. **count(distinct col) 计算该列除 NULL 之外的不重复数量。注意 count(distinct col1, col2) 如果其中一列全为 NULL，那么即使另一列有不同的值，也返回为 0。**
2. **当某一列的值全是 NULL 时，count(col)的返回结果为 0，但 sum(col)的返回结果为NULL，因此使用 sum()时需注意 NPE 问题。**

正例：可以使用如下方式来避免 sum 的 NPE 问题：SELECT IF(ISNULL(SUM(g)),0,SUM(g)) FROM table;

1. 使用 ISNULL()来判断是否为 NULL 值。注意：NULL 与任何值的直接比较都为 NULL。

说明：

1） NULL<>NULL 的返回结果是 NULL，不是 false。

2） NULL=NULL 的返回结果是 NULL，不是 true。

3） NULL<>1 的返回结果是 NULL，而不是 true。

1. 在代码中写分页查询逻辑时，若 count 为 0 应直接返回，避免执行后面的分页语句。
2. **不得使用外键与级联，一切外键概念必须在应用层解决。**

说明：（概念解释）学生表中的 student\_id 是主键，那么成绩表中的 student\_id 则为外键。如果更新学生表中的 student\_id，同时触发成绩表中的 student\_id 更新，则为级联更新。外键与级联更新适用于单机低并发，不适合分布式、高并发集群；级联更新是强阻塞，存在数据库更新风暴的风险；外键影响数据库的插入速度。

1. 禁止使用存储过程，存储过程难以调试和扩展，更没有移植性。
2. in 操作能避免则避免，若实在避免不了，需要仔细评估 in 后边的集合元素数量，控制在 1000 个之内。

# ORM 规约

1. **在表查询中，一律不要使用 \* 作为查询的字段列表，需要哪些字段必须明确写明。**

说明：1）增加查询分析器解析成本。2）增减字段容易与 resultMap 配置不一致。

1. **POJO 类的 boolean 属性不能加 is，而数据库字段必须加 is\_，要求在 resultMap 中进行字段与属性之间的映射。**

说明：参见定义 POJO 类以及数据库字段定义规定，在 sql.xml 增加映射，是必须的。

1. 不要用 resultClass 当返回参数，即使所有类属性名与数据库字段一一对应，也需要定义；反过来，每一个表也必然有一个与之对应。

说明：配置映射关系，使字段与 DO 类解耦，方便维护。

1. **xml 配置中参数注意使用：#{}，#param# 不要使用${} 此种方式容易出现 SQL 注入。**
2. 应用到ibatis的项目，所有的数据库操作（增、删、改、查）必须手动在sqlmap文件中写sql语句

说明：

1. 不允许直接拿 HashMap 与 HashTable 作为查询结果集的输出。

反例：某同学为避免写一个<resultMap>，直接使用 HashTable 来接收数据库返回结果，结果出现日常是把 bigint 转成 Long 值，而线上由于数据库版本不一样，解析成 BigInteger，导致线上问题。

1. **更新数据表记录时，必须同时更新记录对应的 gmt\_modified 字段值为当前时间。**
2. 不要写一个大而全的数据更新接口，传入为 POJO 类，不管是不是自己的目标更新字段，都进行 update table set c1=value1,c2=value2,c3=value3; 这是不对的。执行 SQL 时，尽量不要更新无改动的字段，一是易出错；二是效率低；三是 binlog 增加存储。
3. @Transactional 事务不要滥用。事务会影响数据库的 QPS，另外使用事务的地方需要考虑各方面的回滚方案，包括缓存回滚、搜索引擎回滚、消息补偿、统计修正等。